**APPENDIX C**

A copy of a PowerPoint presentation by Kenneth Yun entitled, "Dynamic Circuits" available on the Internet at http://paradise.ucsd.edu/class/ece165/notes/lec7.pdf is enclosed as an example of evaluating digital logic signals when an input signal transitions monotonically.

# Dynamic Circuits



Kenneth Yun
UC San Diego

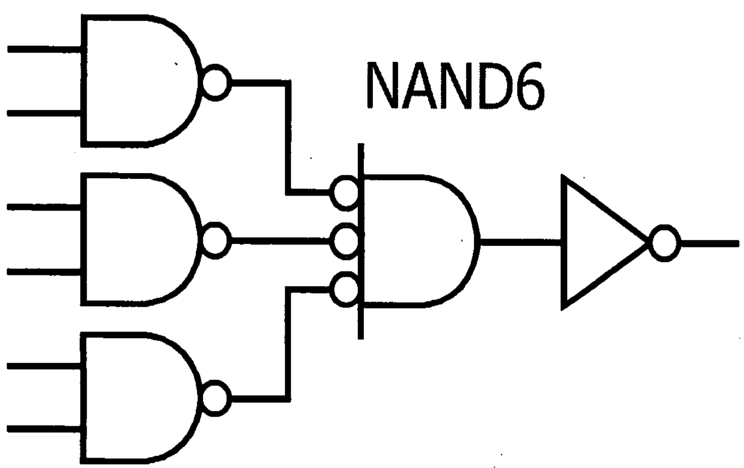
Adapted from EE271 notes,
Stanford University

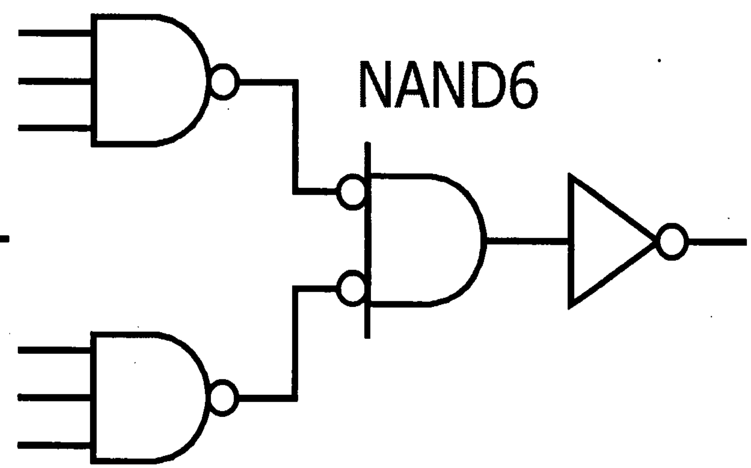
# Overview

- Pseudo nMOS
- Precharged logic
- Domino logic
- Dual-rail logic
- Circuit optimization
- Reading
  - W&E 5.4

# Problems with CMOS: Large Fanin

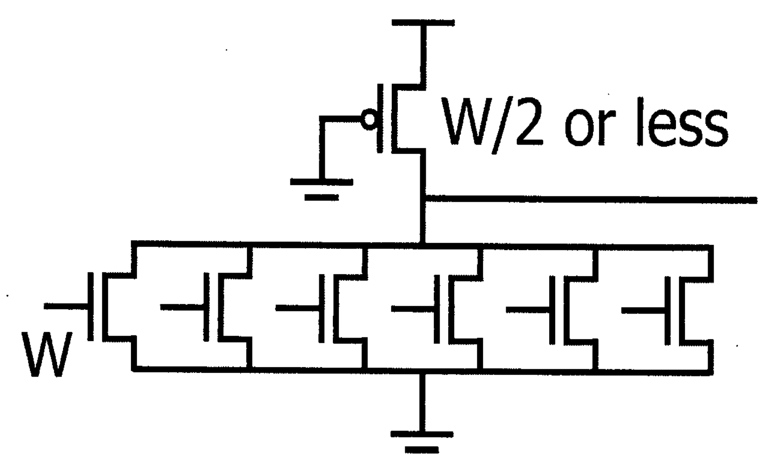
- All CMOS gates require series stack
  - Number of transistors in the series stack is usually equal to the number of inputs
  - Since the series stack is slow, must limit fanin to 3 or 4
- How are large fanin gates built?
  - Use fanin tree



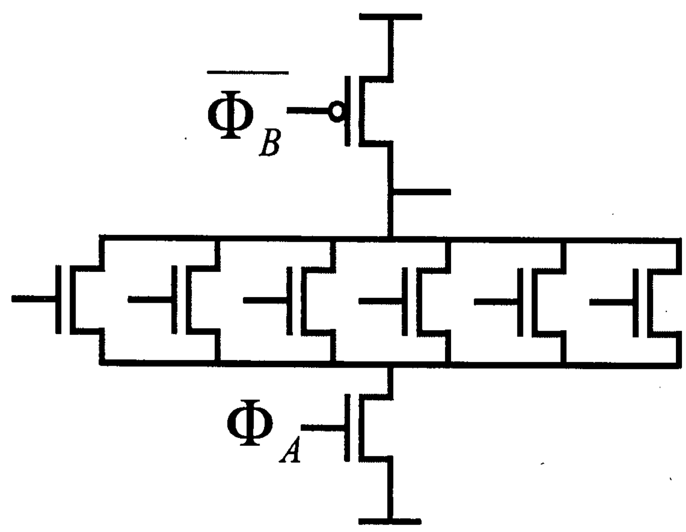
# Pseudo nMOS

- Similar to nMOS, except depletion mode transistor replaced with pMOS (with its gate grounded)
- As in nMOS, problems with
  - DC power dissipation
  - Ratio rule (for 4:1 resistance ratio, $W_n=2W_p$)

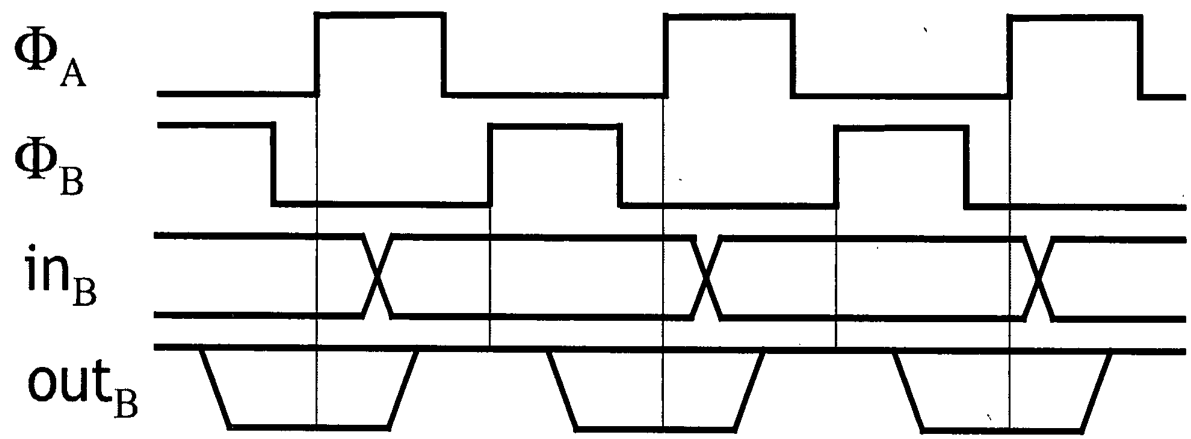


# Precharged Gate

- Precharge output to 1
- Discharge only if output needs to be 0
- Large fanin gates are possible
- No static power dissipation



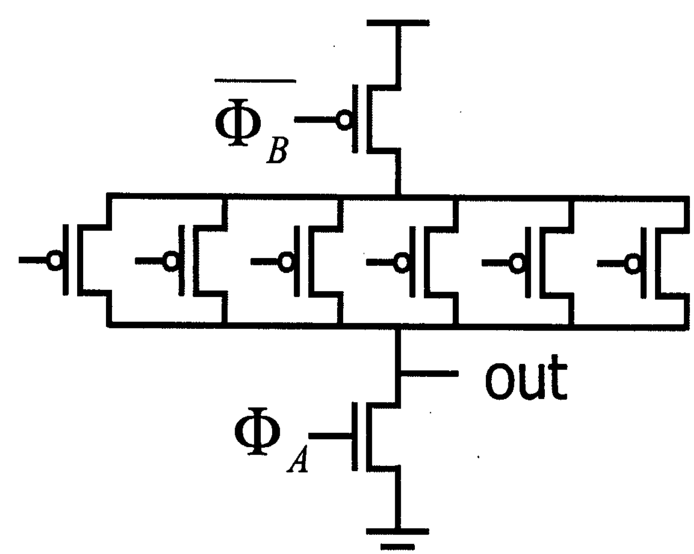
# Precharged Gate Timing




- $out_B$ precharged high when $\Phi_A$ high
- evaluates (falls monotonically or remains high) when $\Phi_B$ high
- $in_B$ must be stable (or rise monotonically) during evaluation of $out_B$

# Predischarged Gate

- Predischarge output to zero when $\Phi_A$ high
- Evaluate (keep it low or pull it up) when $\Phi_B$ high
- Functions correctly but slower than precharged logic (because ?)



# Speed of Pre(dis)charged Logic

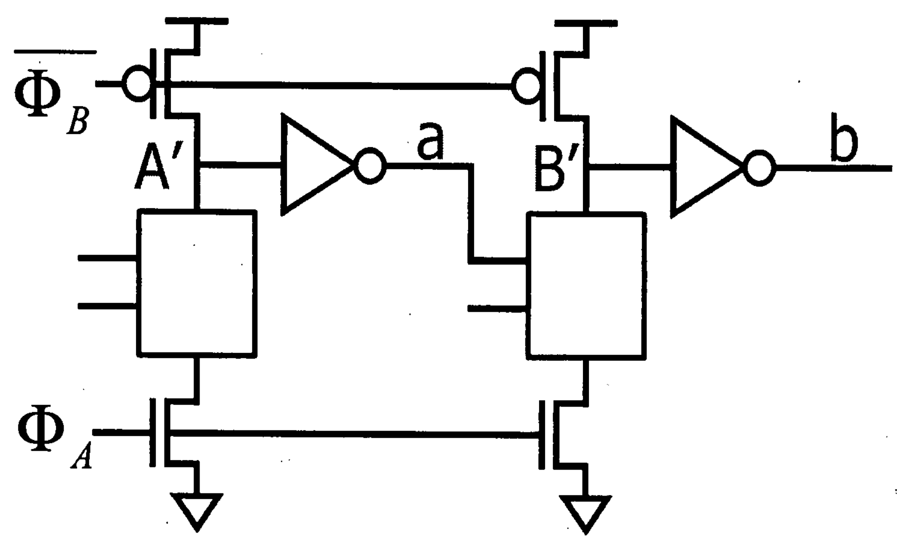
- Precharged logic
  - NOR fast
  - NAND slow
    - Require series stack
- Predischarged logic
  - NAND fast
  - NOR slow
    - Require series stack

# Pros and Cons of Precharged Logic

- Advantages
  - Fast
    - Less gate loading (only half the transistors)
    - Only need to worry about speed of one transition (can make eval transistor bigger)
  - Dense
    - Only need to build pull-down trees
- Disadvantages
  - Inputs must be monotonically rising
    - To drive another precharged gate, output must be inverted (using a static inverter)

# Domino Logic

- Cascade of precharged and static gates pairs
  - a and b become low, when A′ and B′ precharged
  - A′ falls monotonically (or remains high) during evaluation, which causes a to rise (or remain low)
  - If a remains low, B′ remains high; however, if a rises, B′ may fall, which in turn causes b to rise.

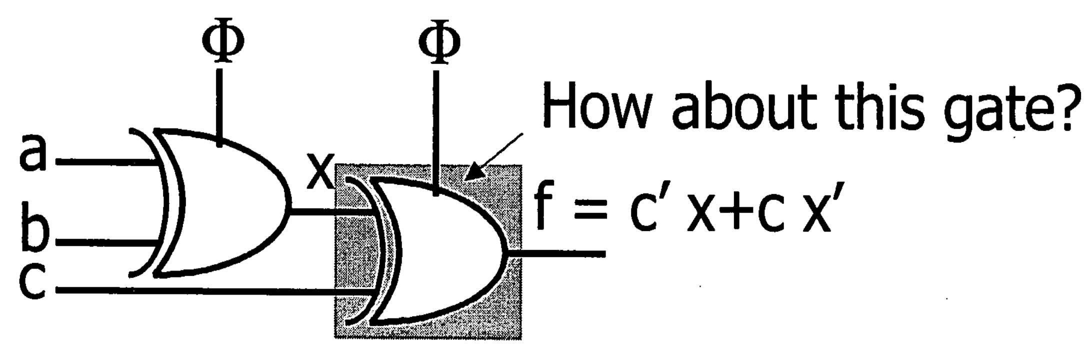


# Domino Logic (Cont’d)

- Called domino because successive falling of A’ and B’ resembles falling domino
- Outputs rise monotonically
- Inputs must rise monotonically (during eval)
    - Falling inputs have no effect on output
    - So, cannot have both a signal and its complement as inputs, unless both have stabilized before eval begins and remain constant during eval. Why?
    - How are non-monotonic functions, such as XOR, built in domino then?

# Domino Logic (Cont'd)

- Clock the gates!
  - The first XOR okay, as long as a, b, a′, and b′ remain stable during eval
- What about the second XOR gate?
  - Requires x and x′ as inputs, which change during evaluation
  - x′ falls during evaluation!

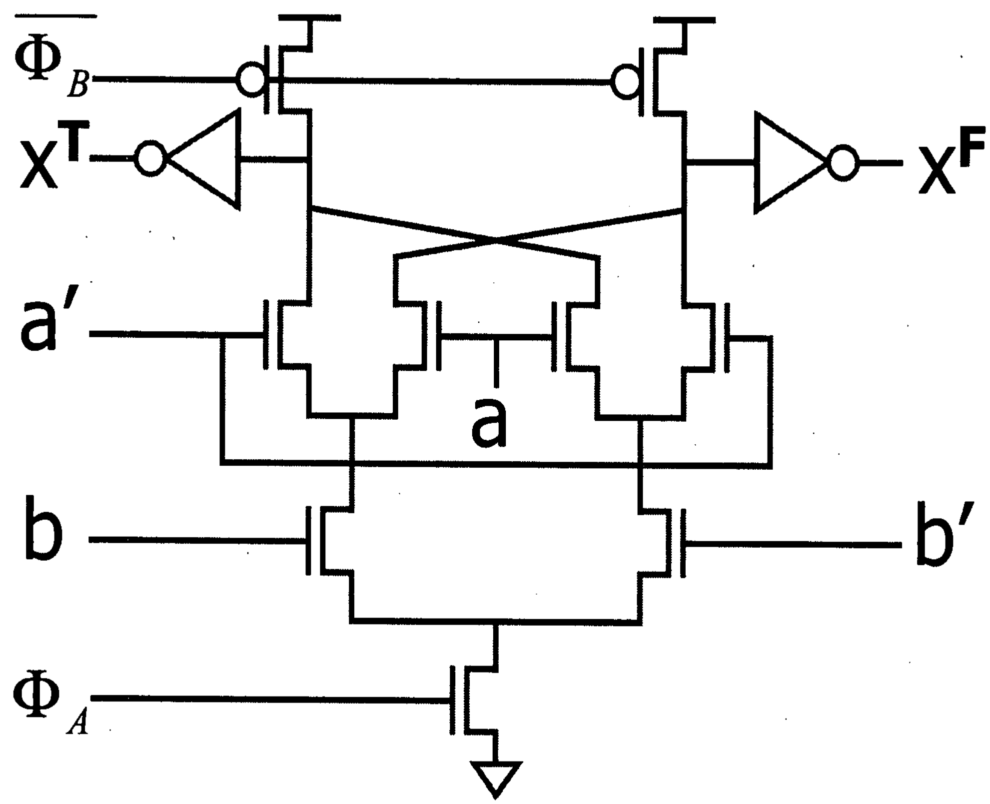


# Solution: Dual-Rail Domino

- Build XOR as dual-rail domino
  - That is, the gate generates both true and complemented versions of output
    - $x^T = a' b + a b'$
    - $x^F = a b + a' b'$
  - Both $x^T$ and $x^F$ are monotonically rising
  - Hence the second XOR gate works correctly
    - $f^T = c' x^T + c x^F$
    - $f^F = c x^T + c' x^F$

# Dual-Rail XOR

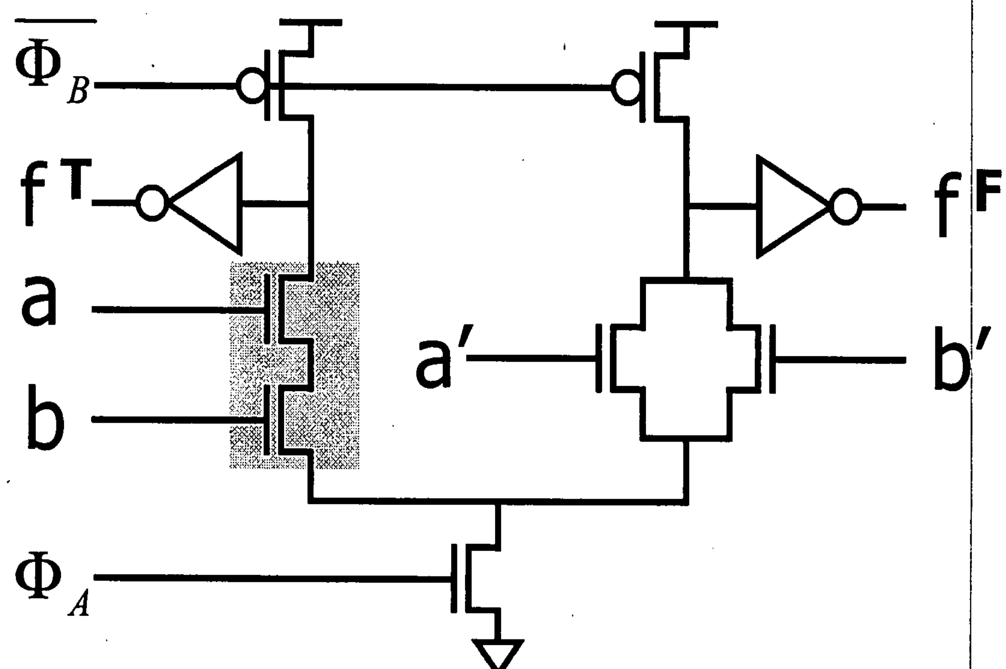
- Merge $x^T$ and $x^F$ in a single gate
- $a^T$, $b^T$, $a^F$, and $b^F$, if ??



# Limitations of Dual-Rail

- Need to build both $f^T$ and $f^F$
  - Parallel transistors in $f^T$ become series stacks in $f^F$ and vice versa
  - Cannot large fanin dual-rail gates
- Most gates, unlike XOR, don't share many transistors

$f^T = (ab)'$
$f^F = ab$

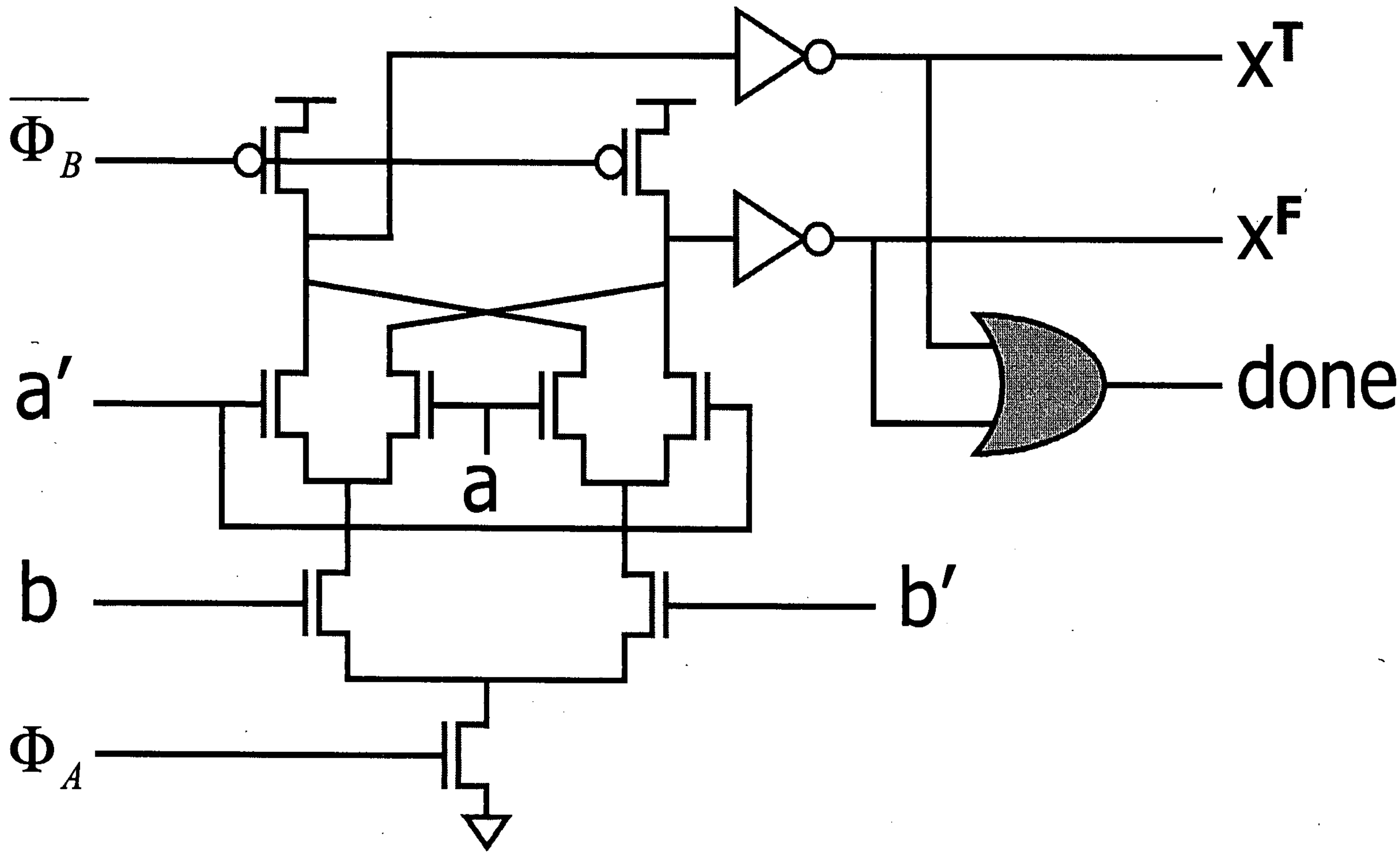


# Dual-Rail Signaling

- Dual-rail gates are complete (can implement any logic function)
- Indicate both value and completion status
  - Before computation completed, both wires are low
  - When completed, exactly one wire goes high
  - Great for self-timed sequencing! Why?

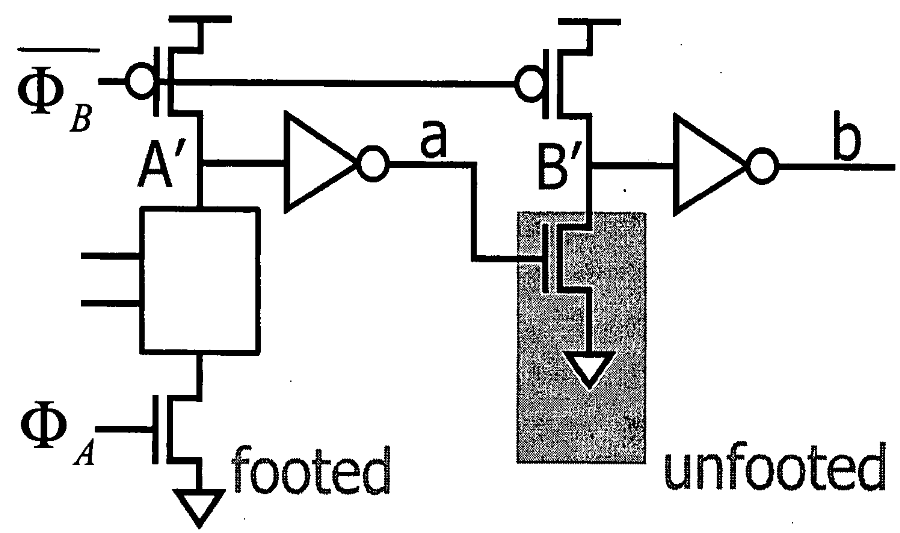
| $a^T$ | $a^F$ | Meaning |
|---|---|---|
| 0 | 0 | Reset (not yet evaluated) |
| 0 | 1 | Eval done with output value '0' |
| 1 | 0 | Eval done with output value '1' |
| 1 | 1 | Cannot occur (error) |

# Dual-Rail Gate w/Completion



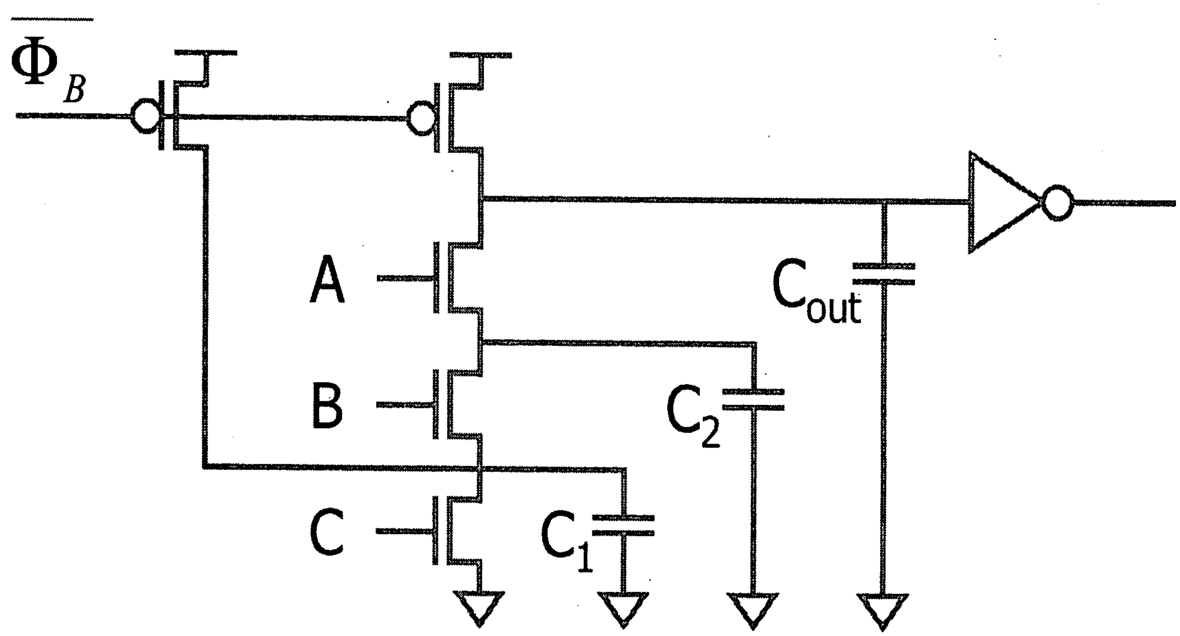
# Unfooted Domino

- If all inputs come from other domino gates, explicit eval transistor not needed
  - Why?
  - Reduces the height of nMOS stack
  - Need to make sure that the first gate's output precharged before precharging the next gate can be precharged. Why?



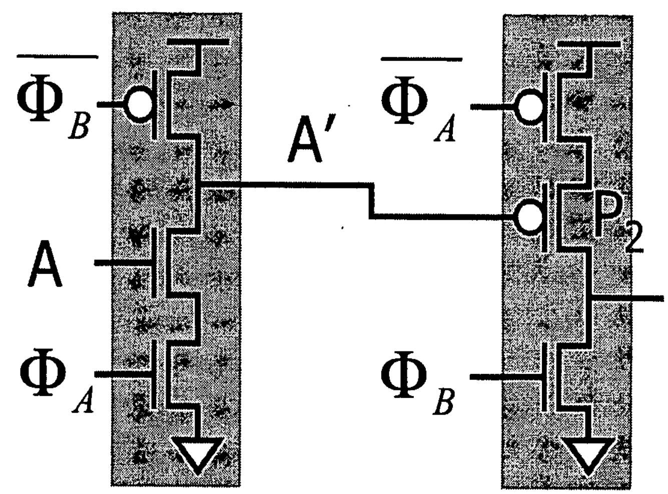
# Charge Sharing in Precharged Gate

- Assume that $C_1$ and $C_2$ had been discharged during last eval and $C_{out}$ precharged
- What happens, during eval, if A and B rise but C remains low?
- What are potential solutions to this problem?



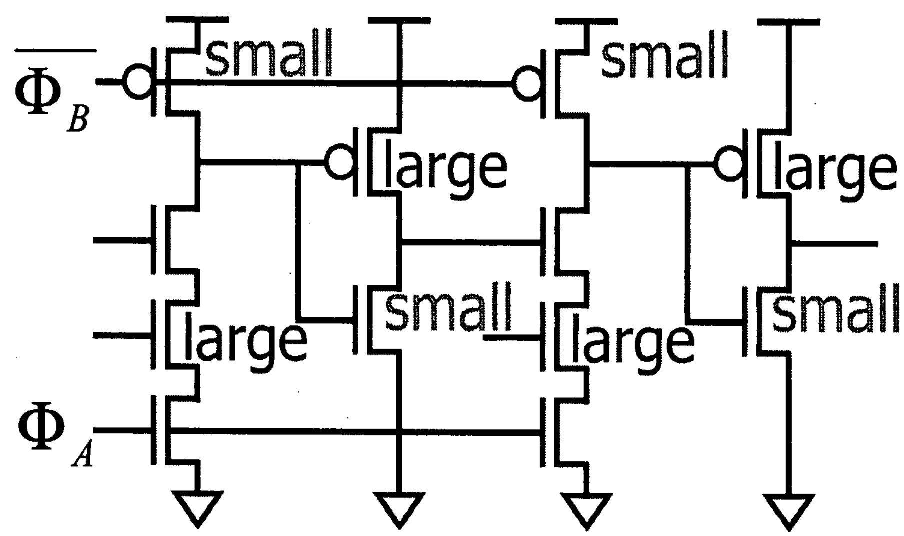
# NORA

- Precharged nMOS gate followed by a precharged pMOS gate
- A′ normally high, keeping $P_2$ off, without requiring an inverter between two stages
- But serious noise margin problems
  - What happens if the signal level on node A′ is degraded?



# Optimizing for Single Edge

- Can improve evaluation speed by
  - making nMOS of precharged gate larger *and*
  - making nMOS of static inverters much smaller than pMOS
- What happens to precharge speed then?
  - Does it matter? Why or why not?



# Clocked AND

- A compact qualified clock generator
  - Precharged AND gate
- Need to worry about clock skew